

# Modeling and Planning in Large State and Action Spaces

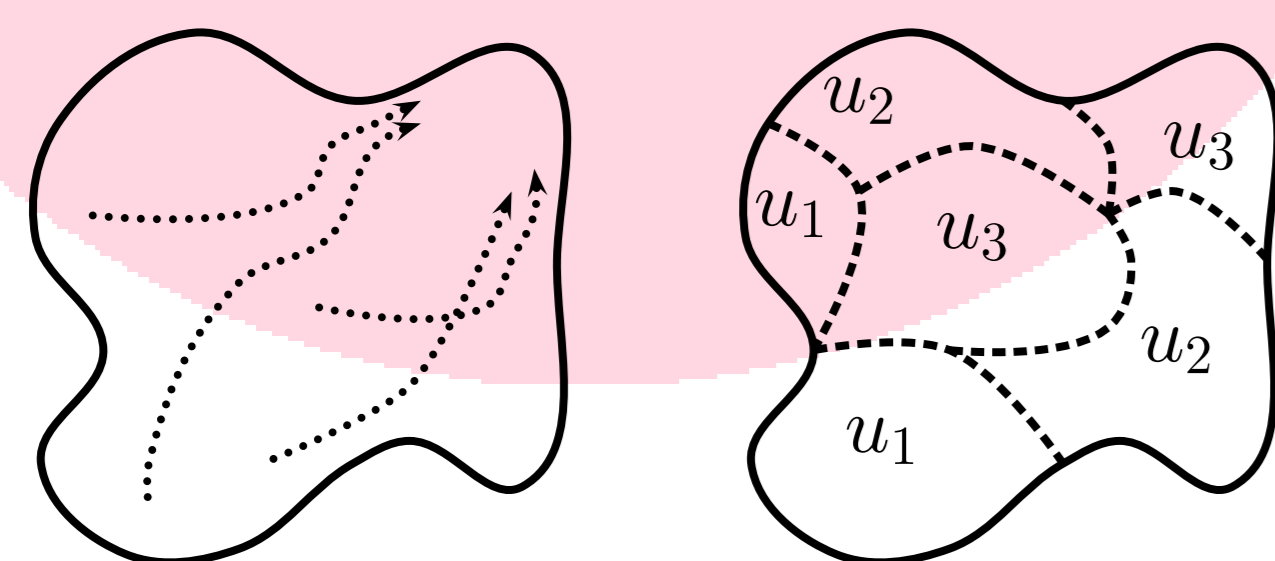
**Mykel J. Kochenderfer and Gillian Hayes**

m.kochenderfer@ed.ac.uk, gmh@inf.ed.ac.uk



## Problem Statement

How can an agent best use its raw experience to build a reactive plan while interacting with the world?

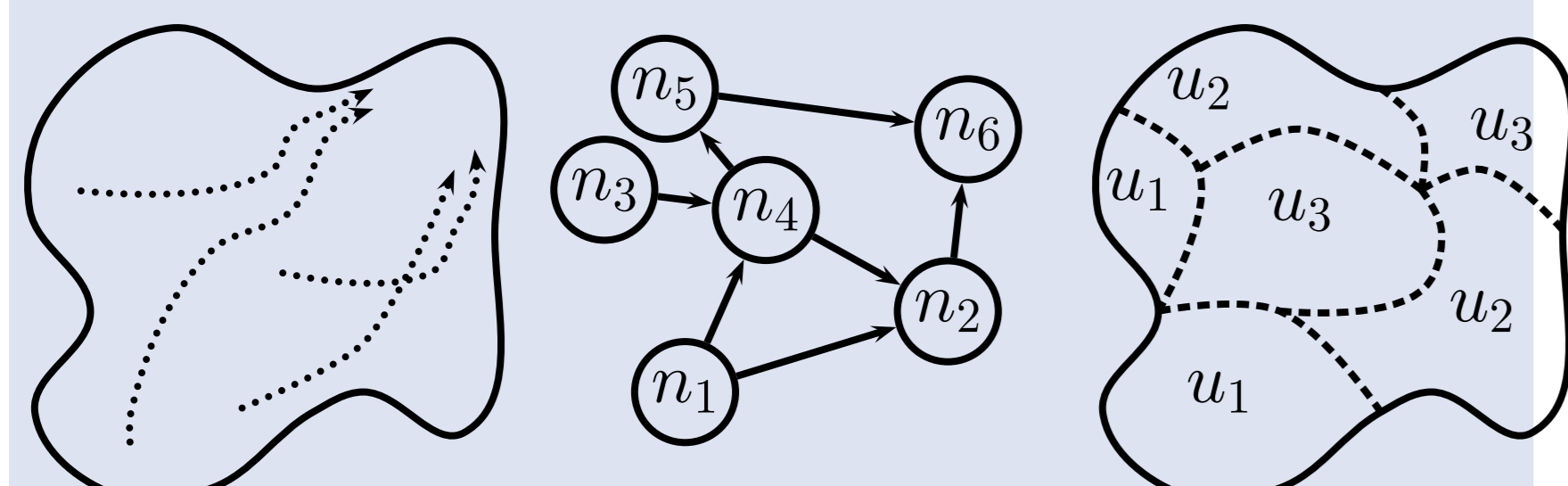


Experience → Plan

There are a number of challenges:

- **Generalization:** State and action spaces are large
- **Managing Uncertainty:** Transitions are stochastic
- **Adaptation:** World dynamics are initially unknown

Our approach is to estimate a world model and then compute the plan that maximizes the expected discounted reward.

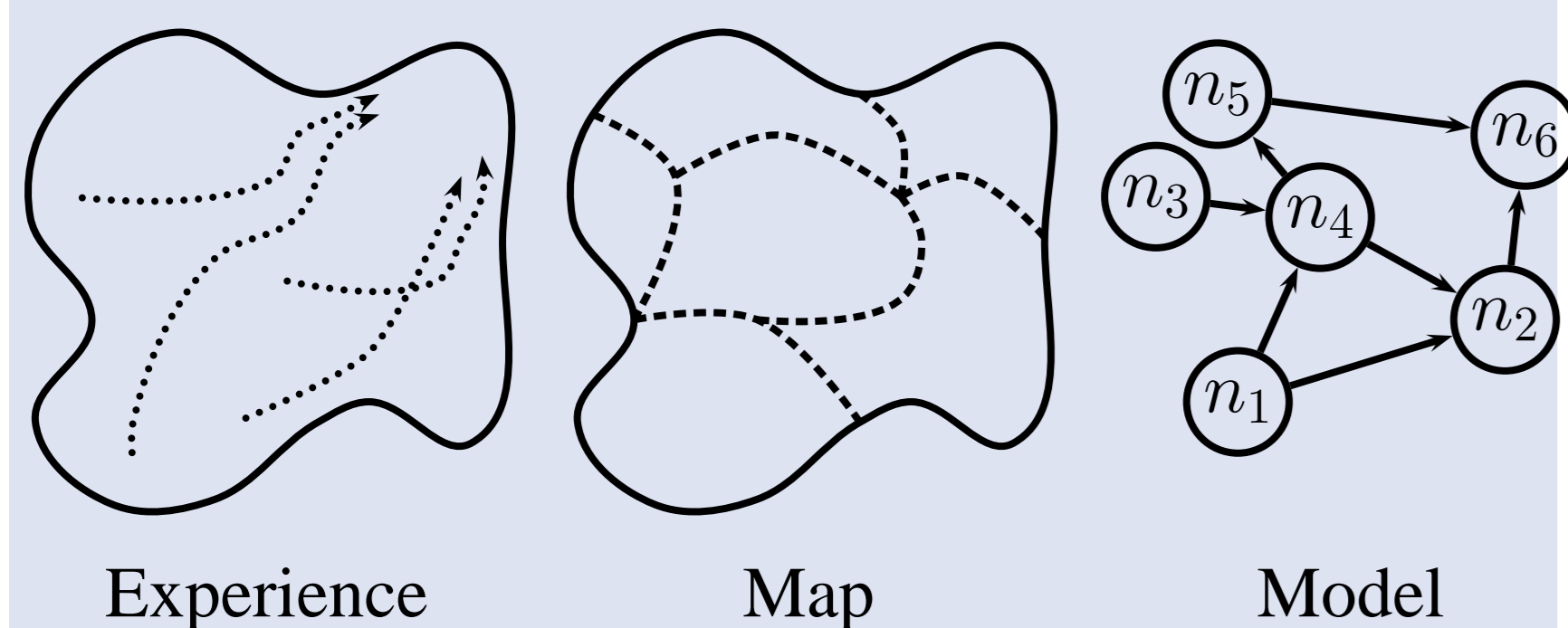


Experience Model Plan

This process is broken into *modeling* and *planning*.

## Modeling

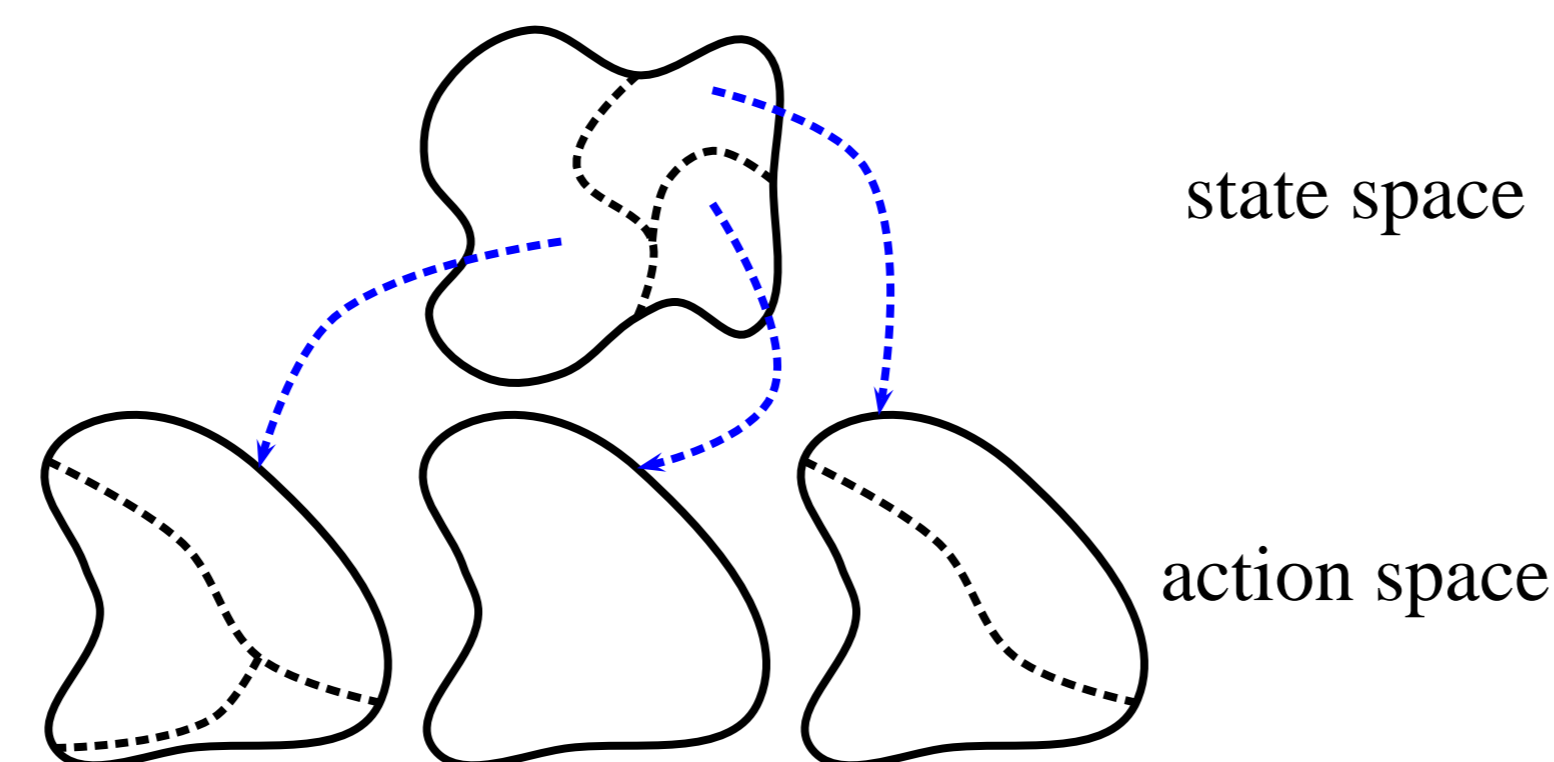
Modeling involves partitioning the state and action spaces into regions and extracting a model (represented as a semi-Markov decision process).



Experience Map Model

Partitioning the state and action spaces from scratch is prohibitively expensive. Instead, we should split and merge already existing regions.

Each region of the state space has its own partitioning of the action space associated with it, as shown below.



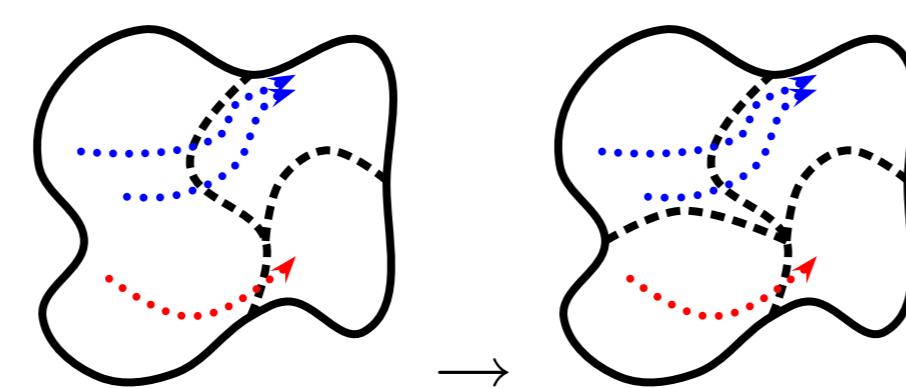
The following operations are defined:

- $SPLIT(n, S_1, \dots, S_m)$
- $SPLIT(u, A_1, \dots, A_m)$
- $MERGE(n_1, \dots, n_m)$
- $MERGE(u_1, \dots, u_m)$

These operations may be implemented using decision trees, nearest neighbor, etc.

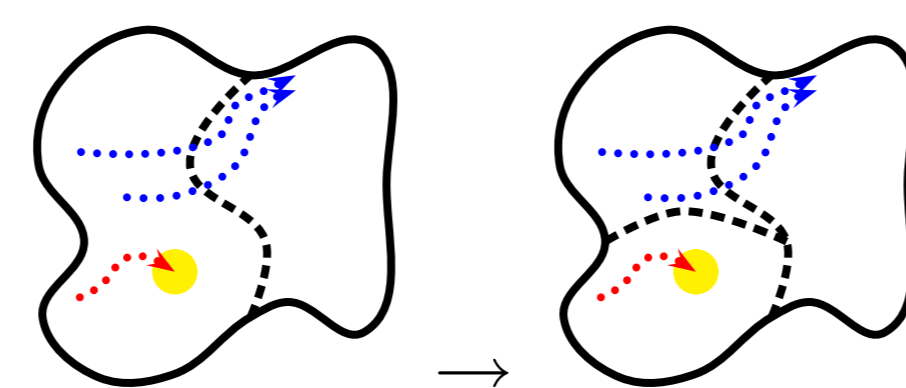
We want to split states and actions that behave differently and merge regions that behave similarly. Previous ideas involved computing estimates of the value function at individual states in the same region and comparing this to the value function of the region using the Komogorov-Smirnov test [1] or sum-squared error [4].

## Transition Splitting



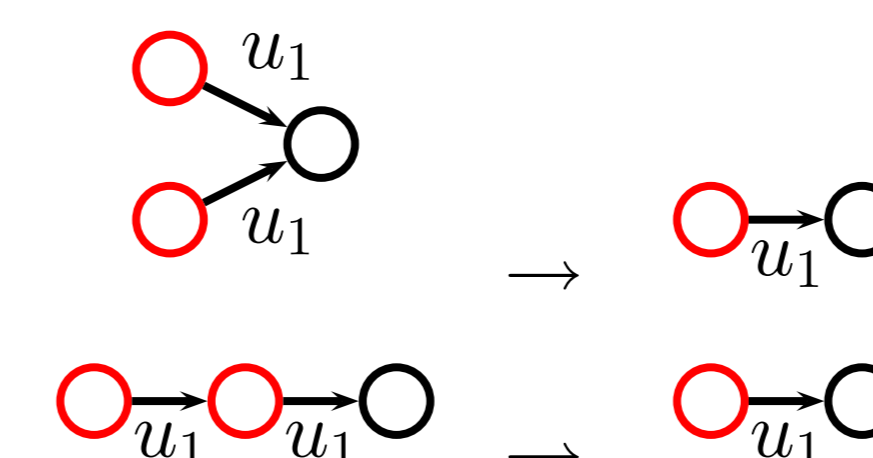
Split states involved in transitions with different expected discounted reward.

## Failure Splitting



Split states involved in successful transitions from those involved in failure.

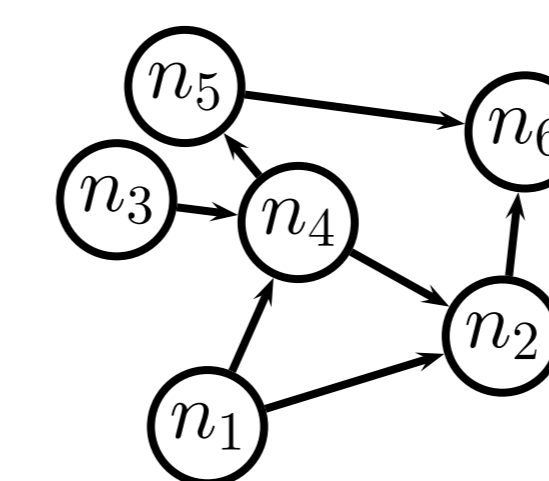
## Merging



Use a process related to the SQUISH algorithm [3].

The agent makes these revisions when it has time. Priority is assigned to different revision operations according to a heuristic.

## Planning

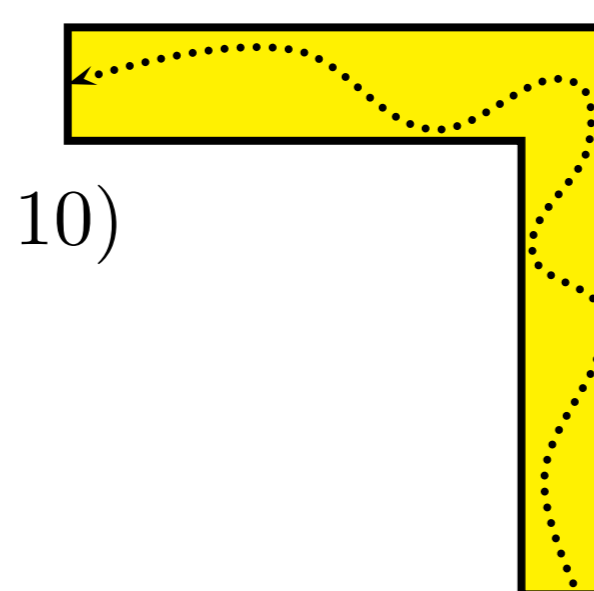


Planning over the model is done incrementally in a manner similar to the dynamic programming process called prioritized sweeping [2]. This process prioritizes updates of the value function over the regions of the state space. When the agent has time, it selects the highest priority region and updates the value function.

## Corner World Experiment

The objective is to steer a holonomic robot from the start line to the finish line as quickly as possible.

- actions:  $\theta \in [0, 2\pi)$
- states:  $\langle x, y \rangle, x \in (0, 10), y \in (0, 10)$
- translation speed: 5 m/s
- sample frequency: 5 Hz
- noise: Gaussian with 5 mm s.d.
- timeout: 100 s



The state and action spaces are partitioned by decision graphs. Regions are split using axis-parallel hyperplanes, chosen according to information gain.

The agent is steered toward the finish line by a 5% noisy teacher. After only two training episodes, the agent is able to construct a model and a competent reactive plan based on its experience. As the agent accumulates new experience, it revises its model and reactive plan.

## Summary

The following diagram summarizes the processes and data structures used to implement this system.

Processes	Data Structures
Map Revision splits and merges regions	Map state and action regions
Experience Revision adds and removes observations	Experience observations
Action Selection chooses best actions	Model dynamics, reward, failure
Plan Revision chooses regions to update	Plan value, greedy action

## References

- [1] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, 1995.
- [2] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, October 1993.
- [3] Nils J. Nilsson. Learning strategies for mid-level robot control: Some preliminary considerations and results. [www.robotics.stanford.edu/users/nilsson](http://www.robotics.stanford.edu/users/nilsson), May 2000. Department of Computer Science, Stanford University.
- [4] William T. B. Uther and Manuela M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 769–775, 1998. AAAI.